Définition et identification des tables de nomenclatures

Julien Delplanque, Olivier Auverlot, Anne Etien, Nicolas Anquetil

Définition et identification des tables de nomenclatures

Julien Delplanque [1], **Olivier Auverlot** [1], **Anne Etien** [1], **Nicolas Anquetil** [1]

Université de Lille, CRIStAL, CNRS, UMR 9189,
RMoD Team, Inria Lille Nord Europe
Lille, France
{prenom}.{nom}@inria.fr,olivier.auverlot@univ-lille1.fr

RÉSUMÉ. *Dans une base de données relationnelle, certaines tables ont pour finalité de rassembler et d'apporter des informations complémentaires aux lignes des tables constituant le coeur de la base. Ces données sont stockées dans des tables que nous désignons "tables de nomenclatures". Pouvoir les distinguer présente de nombreux intérêts dans le cadre de l'étude, la maintenance et l'évolution d'une base de données. Nous proposons des propriétés permettant de définir la nature de ces tables. Une expérience permettant de valider les propriétés proposées est décrite puis appliquée à un cas d'étude. Un modèle de classification pour les tables de nomenclatures est construit à l'aide d'un algorithme d'exploration de données (datamining). Son évaluation montre une précision* $88,6\%$ *et un rappel de* $88,7\%$.

ABSTRACT. *In a relational database, some tables are used to gather additional information to rows of tables forming the core of the database. This data is stored in tables that we call "nomenclature tables". Being able to distinguish them offers many interests in the study, maintenance and evolution of databases. We propose properties to define the nature of these tables. Then, an experiment to validate the proposed properties is described and applied on a case study. A classification model for nomenclature tables is built using a datamining algorithm. Its evaluation shows a precision of* 88.6% *and a recall of* 88.7%.

MOTS-CLÉS : *table de nomenclatures, évolution des données, base de données relationnelle*

KEYWORDS: *nomenclature table, data evolution, relational databases*

DOI:10.3166/RIA.28.1-**??**

1. Introduction

Souvent évoquée par les architectes de base de données ou les programmeurs, la notion de table de nomenclatures est pourtant un concept flou issu de la connaissance empirique. Pour définir l'expression " table de nomenclatures ", il est judicieux de se pencher sur la définition même du terme nomenclature. Dans le dictionnaire " Le petit Robert " (*Le Nouveau Petit Robert de la langue française 2009*, 2008), ce mot est défini comme étant *" l'ensemble des termes employés dans une science, une technique, un art..., méthodiquement classés; une méthode de classement de ces termes. C'est également l'ensemble des formes (mots, expressions, morphèmes) répertoriées dans un dictionnaire, un lexique et faisant l'objet d'un article distinct "*.

Trois idées fortes émergent de cette définition. Une nomenclature est une liste d'éléments ayant une cohérence thématique. Cette collection est structurée pour faciliter la recherche d'un terme. Chacun de ces termes est unique afin d'être identifiable.

Le but de cet article est de définir ce qu'est une table de nomenclatures de façon plus précise afin de pouvoir, dans le futur, aider les architectes de base de données à les reconnaitre à l'aide d'un outil logiciel. Le reste de cet article est organisé comme suit. Dans la Section 2, nous présentons l'intérêt de différencier les tables de nomenclatures des autres tables. La Section 3 propose une définition pour le concept de nomenclatures et décrit une expérience pour tester sa validité. La Section 4 expose un cas d'étude pour vérifier la définition proposée dans la Section 3. La Section 5 positionne cet article par rapport à la littérature. La Section 6 conclut cet article et propose des pistes à explorer dans le cadre de travaux futurs.

2. Motivations

Pour faciliter l'évolution des systèmes d'information basés sur des bases de données relationnelles, distinguer les tables de nomenclatures permet de mettre en avant les tables constituant le noyau du domaine métier et faciliter ainsi sa compréhension. Il s'agit d'éliminer le bruit dû aux tables dont l'analyse n'est pas essentielle pour appréhender le fonctionnement général du schéma. L'objectif est de réduire le nombre de tables et donc le temps d'analyse.

Une table de nomenclatures est, a priori, utilisée majoritairement en lecture seule. Face aux besoins croissants de performances, il est primordial de réduire les temps d'accès aux informations. Identifier les tables de nomenclatures permet de détecter les tables nécessitant une méthode d'indexation optimisée pour la lecture des données.

Afin de garantir la sureté de fonctionnement d'un système d'information, il est important de localiser l'utilisation de littéraux dans le code SQL. Ils sont généralement utilisés comme condition de selection dans les clause `WHERE` des requêtes SQL. L'utilisation de ces littéraux peut être considérée comme une mauvaise pratique car s'il s'agit d'une valeur numérique correspondant à la valeur d'une clé primaire, le code sera peu lisible. Il sera nécessaire de se référer au contenu de la table cible pour connaitre la sémantique de la requête. Si le littéral correspond à une donnée contenue

dans la ligne (hors clé primaire), toute modification de cette information dans la table nécessitera la modification de l'ensemble des requêtes SQL utilisant ce littéral. Dans une condition de sélection, si l'une des deux parties de l'expression est une référence à une clé étrangère et que l'autre partie est un littéral, il est probable que la table ciblée par la clé étrangère soit une table de nomenclatures. En effet, si l'architecte utilise un littéral pour référencer une ligne, il fait l'hypothèse que la ligne correspondante ne peut être supprimée et que la valeur du littéral ne peut être altérée. Si la valeur concrète de la clé primaire ne correspond pas à celle exprimée dans le code client, l'application produira un résultat erroné ou ne pourra pas fonctionner.

Dans un environnement Cloud et afin de répondre aux problématiques du provisioning, la structure et les informations d'une base de données peuvent être réutilisées dans leur intégralité ou en partie pour construire une nouvelle instance de la base originale. Distinguer une table métier d'une nomenclature permet d'opter pour une stratégie d'initialisation optimale. Le contenu d'une table métier ne sera pas dupliqué. Le contenu d'une table de nomenclatures sera totalement ou partiellement préservé.

3. Proposition

Nous souhaitons proposer des critères pour distinguer les tables de nomenclatures des autres tables de la base de données. À partir de ces critères, nous proposons une expérience pour déterminer si ceux-ci décrivent correctement les nomenclatures.

3.1. Propriétés

En reprenant les éléments issus de la définition du terme nomenclature et des cas d'utilisation cités précédemment, nous proposons 5 propriétés pour reconnaitre les nomenclatures parmi l'ensemble des tables d'une base de données.

1. Chaque ligne d'une table de nomenclatures est identifiable de façon unique. En utilisant le modèle relationnel, cela se traduit par l'application d'une contrainte de clé primaire sur une ou plusieurs des colonnes de la table.

2. Chaque ligne d'une table de nomenclatures est unique. Cela implique qu'en excluant la clé primaire de l'analyse, que deux lignes d'une même table, si on les compare colonne à colonne, ne peuvent pas être identiques.

3. Une table de nomenclatures est souvent référencée par d'autres tables et référence rarement des tables via des contraintes de clés étrangères.

4. Une table de nomenclatures évolue peu. Si elle évolue, il s'agit majoritairement d'ajouts de lignes, rarement de modifications et encore plus rarement de suppressions.

De par sa fonction de glossaire, une table de nomenclatures a pour rôle de décrire des données en fournissant des informations complémentaires sur celles-ci. Elle entretient donc une relation limitée et encadrée avec une ou plusieurs tables métiers. Il est essentiel, dans une base de données relationnelle, de maintenir la cohérence des données et donc le bon fonctionnement des requêtes SQL. Nous pouvons donc en dé-

duire que la suppression d'une ligne dans une table métier ne doit pas être induite par la suppression d'une ligne dans une table de nomenclatures. De plus, la suppression d'une ligne dans une table de nomenclatures ne peut être autorisée si la clé primaire de la ligne est référencée une ou plusieurs fois via une clé étrangère d'une table métier. Enfin, si la clé primaire d'une table de nomenclatures est modifiée, sa nouvelle valeur doit être mise à jour dans les clés étrangères la référençant dans les tables métiers. Ces observations nous amènent à énoncer une cinquième et dernière propriété :

5. Certaines contraintes d'intégrité référentielle sont compatibles avec la notion de table de nomenclatures et d'autres non.

3.2. Description de l'expérience

L'objectif de l'expérience est de déterminer s'il est possible de distinguer les tables de nomenclatures des autres tables via l'analyse des propriétés décrites dans la section précédente. Cette expérience se déroulera en 3 étapes décrites en détail ci-dessous.

3.2.1. Classification par l'architecte

La première étape de l'expérience consiste à demander au concepteur de la base de données de classer les tables en deux catégories. La première regroupe les tables métiers dont la présence est essentielle pour que la base de données puisse remplir sa mission. La seconde est constituée des tables de nomenclatures. Cette répartition est l'oracle pour évaluer les méthodes de classifications utilisées dans l'expérience.

3.2.2. Extraction des métriques

La seconde étape de l'expérience consiste en l'extraction de métriques pour chaque table de la base de données. Certaines métriques concernent la dernière version de la base de données et d'autres se basent sur un historique de plusieurs versions. Les métriques suivantes se calculent à partir du résultat de la différence entre les paires de versions consécutives de la base de données. Pour une différence entre la version i et la version j, chaque table dispose des métriques suivantes :

- $t^{+}_{i,j}$ est le nombre de lignes ajoutées,
- $t^{-}_{i,j}$ est le nombre de lignes supprimées,
- $t^{\sim}_{i,j}$ est le nombre de lignes modifiées,
- t_i et t_j sont les nombres de lignes respectivement dans les version i et j.

Pour déterminer le nombre de lignes ajoutées, supprimées ou modifiées, la méthode se base sur la clé primaire définie dans le schéma SQL afin d'identifier chaque ligne. Pour trouver les différences entre deux versions d'une même ligne, le contenu est analysé colonne par colonne. À partir de ces métriques extraites via l'analyse des versions de la base de données, d'autres métriques ont été calculées. Soit n versions numérotés de 0 à $n-1$, pour chaque table, les métriques sont les suivantes:

– $N = t_0 + \sum_{i=0}^{n-2} t^+_{i,i+1}$ est le nombre total de lignes ayant existé dans une table,

– $T^+ = \frac{\sum_{i=0}^{n-2} t^+_{i,i+1}}{N}$ est le ratio de lignes ajoutées dans une table sur toutes les versions du schéma par rapport au nombre total de lignes ayant existé dans cette table,

– $T^- = \frac{\sum_{i=0}^{n-2} t^-_{i,i+1}}{N}$ est le ratio de lignes supprimées dans une table sur toutes les versions du schéma par rapport au nombre total de lignes ayant existé dans cette table,

– $T^\sim = \frac{t^\sim_{0,n-1}}{t_{n-1}}$ est le ratio de lignes modifiées entre la première et la dernière version du schéma divisé par le nombre de lignes dans la dernière version du schéma.

La métrique d mesure le nombre de lignes dupliquées pour chaque table de la dernière version du schéma en ne tenant compte que des données stockées dans les colonnes qui ne font pas partie de la clé primaire. Tenir compte de ces colonnes rendrait caduque toute possibilité de trouver une ligne dupliquée.

Les métriques $\#PK$ et $\#FK$ portent sur le nombre de clés primaires (0 ou 1) et de clés étrangères présentes dans chaque table de la dernière version de la base.

L'analyse des contraintes d'intégrité référentielle constitue une dernière métrique. Celles-ci définissent l'action effectuée lors de la modification / suppression d'une valeur référencée par une clé étrangère. Le tableau 1 recense les différentes configurations d'une contrainte d'intégrité référentielle compatibles avec une relation associant une table métier et une table de nomenclatures. Les besoins sont de :

– Interdire la modification de la valeur de la clé primaire dans la table de nomenclatures si cette valeur est référencée dans au moins une des clés étrangères de la table métier. Il s'agit de s'assurer ici que chaque clé étrangère référencie bien une clé primaire de la nomenclature,

– Eviter l'effacement d'une ligne dans la nomenclature qui pourrait laisser des lignes orphelines dans la table métier,

– Faire en sorte que la suppression d'une ligne dans la nomenclature n'entraîne pas l'effacement de la ou des lignes associées dans la table métier,

– Maintenir l'apport sémantique de la nomenclature à la table métier en la préservant de l'affectation de valeurs nulles (`NULL`) dans ses clés étrangères.

Tableau 1. Contraintes d'intégrités compatibles avec une table de nomenclatures.

	UPDATE	DELETE
NO ACTION / RESTRICT	X	X
CASCADE	X	
SET NULL		

3.2.3. Validation de la définition

Cette dernière étape est la validation de la définition de table de nomenclatures via l'analyse combinée des métriques extraites. Un algorithme de datamining est utilisé afin de construire un modèle pour trouver les tables de nomenclatures. L'objectif est d'obtenir un modèle de classification avec une précision et un rappel satisfaisants.

4. Cas d'étude

Nous disposons d'AppSI, une base de données PostgreSQL conçue et utilisée par le Pôle Informatique et Technique du Laboratoire CRIStAL (Université de Lille). AppSI est composée de 97 tables, 63 vues, 64 fonctions PL/pgSQL et de 20 triggers. Cette base de données présente la particularité d'être historisée (peu d'entrées sont donc effacées). Les données mises à notre disposition sont constituées de 21 extractions contenant le schéma SQL et les données anonymisées. Elles couvrent la période allant du 10 novembre 2017 au 13 mars 2018. Le délai entre les différentes extractions n'est pas régulier. Elles correspondent à des sauvegardes réalisées à l'aide de la commande `pg_dump` préalablement à une évolution du schéma SQL.

4.1. Observations sur la dernière version d'AppSI

Selon l'architecte d'AppSI, la base contient 65 tables métiers et 32 tables de nomenclatures. Sur la dernière extraction de la base de données, la métrique d est égale à 0 pour toutes les tables et indique donc qu'aucune table n'a de lignes dupliquées. De plus, seules 2 tables n'ont pas de contraintes de clé primaire. Après discussion avec l'architecte, nous apprenons que cela est dû à des oublis lors de la création de ces tables. La figure 1 montre la répartition des tables en fonction du nombre de clés étrangères définies dans chacune d'entre-elles. Ce graphique permet de remarquer que :

1. Sur les 46 tables sans clé étrangère, 29 sont citées comme nomenclature,
2. Sur les 15 tables ayant une clé étrangère, 2 sont citées comme nomenclature,
3. Une seule table ayant 2 clés étrangères est citée comme nomenclature.

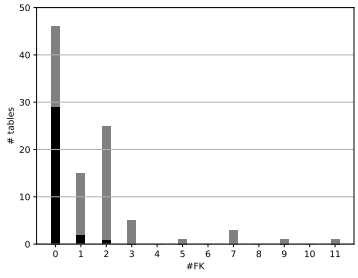


Figure 1. Nombre de tables ayant un certain nombre de clés étrangères. En noir, les tables désignées comme tables de nomenclatures par l'architecte.

En utilisant la propriété 5 concernant les contraintes d'intégrité référencielle, les tables sont classées avec une précision de 81% et un rappel de 71%. 83 tables sont correctement classifiées (26 tables de nomenclatures et 57 tables métier) et 14 ne le sont pas. Parmi ces dernières, 6 tables de nomenclatures et 8 tables métier sont mal catégorisées. Après discussion avec l'architecte, il apparait que les contraintes d'intégrité référencielle sur les 6 tables de nomenclatures ont été mal définies. Les 8 tables métier incorrectement classées sont plus problématiques car si leur définition répond bien à la propriété 5, elles n'en sont pas moins des tables métier.

Il semblerait donc que les propriétés 1, 2, 3 et 5 proposées dans la sous-section 3.1 soient respectées par les tables de nomenclatures. Néanmoins, la propriété 1 ne permet pas de discriminer les nomenclatures des autres tables mais élimine 2 tables qui ne sont pas des nomenclatures. La propriété 2 ne permet pas de discriminer les nomenclatures dans le cas d'AppSI car aucune ligne dupliquée n'a été trouvée dans une table (cette configuration est peut-être spécifique à AppSI). La propriété 3 concernant les clés étrangères semble utile pour différencier les tables de nomenclatures des autres tables. En effet, plus de la moitié des tables n'ayant pas de clés étrangères sont des nomenclatures. Néanmoins, ce critère n'est pas suffisant pour les distinguer. 17 tables sur les 46 trouvées ne sont pas catégorisées comme nomenclature par l'architecte. La propriété 5 fournit de bons résultats mais n'est pas suffisante utilisée seule.

4.2. Observations sur l'historique des versions d'AppSI

L'extraction des métriques concernant l'historique de l'évolution des données n'a de sens que pour les tables dont le contenu a été modifié durant la période d'observation. Nous avons donc retiré les tables n'ayant subi aucun changement. La figure 2 illustre le nombre de tables ayant ou n'ayant pas évolué. Au sein de ces deux groupes et suivant la catégorisation de l'architecte, nous distinguons également les tables supposées de nomenclatures (en noir) des autres (en gris). Nous pouvons constater que sur les 57 tables sans évolution, 35 ont été classées dans les nomenclatures. Parmi les 40 tables ayant évolué, 10 ont été classées dans les nomenclatures.

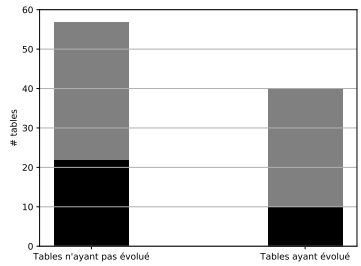
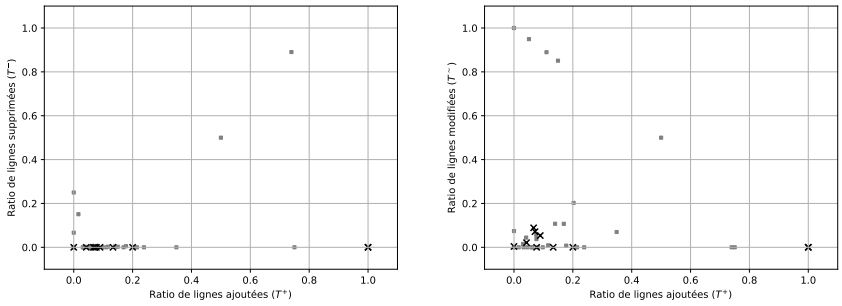


Figure 2. Distinction des tables selon l'observation ou non d'au moins une évolution (ajout, suppression ou modification de ligne entre la première et dernière version de la table). Les parties noires des barres comptabilisent les tables désignées par l'architecte de base de données comme étant des tables de nomenclatures.

La figure 3.1 représente les métriques T^+ et T^- et la figure 3.2 représente les métriques T^+ et $T^{\sim}$ pour les tables ayant évolué. Les nomenclatures indiquées par l'architecte apparaissent en noir sur les graphes. Pour le graphe 3.1, on peut observer que peu de tables ont les deux ratios T^+ et T^- supérieurs à 0 simultanément. Les tables classées parmi les nomenclatures subissent uniquement des ajouts (aucune table classée comme nomenclature n'a sa métrique $T^- > 0$). Deux tables classées comme nomenclature ont 1 comme valeur de métrique T^+ : il s'agit de tables créées et alimentées une seule fois durant la période d'observation. Pour le graphe 3.2, on peut observer que les tables classées comme nomenclatures ont toutes leurs valeurs de métrique $T^{\sim}$ inférieures à 10% ($8,8\%$ pour la table ayant la valeur la plus élevée).



(1) (2)

Figure 3. Pour chaque table durant l'évolution de la base de données : (1) La métrique T^- par rapport à T^+, (2) la métrique $T^{\sim}$ par rapport à T^+. Selon la classification de l'architecte d'AppSI, une table métier est représentée par un carré gris et une table de nomenclatures est représentée par une croix noire.

L'étude de l'évolution des données d'AppSI nous permet de discuter la propriété 4 proposée dans la sous-section 3.1. L'évolution respecte la propriété 4. En effet, les nomenclatures semblent peu évoluer avec majoritairement des ajouts, peu de mise à jour et, aucune suppression. Néanmoins, s'il semble que ces critères d'évolution soient des conditions nécessaires pour classer une table parmi les nomenclatures, ceux-ci restent insuffisants. D'autres tables ont des caractéristiques similaires aux tables classées par l'architecte comme nomenclatures mais n'en sont pas. De plus, l'application des métriques construites pour décrire l'évolution des données de chaque table nous a permis de mettre en avant une faiblesse de la métrique T^+ : si une table est créée durant la période d'observation de la base, cette métrique sera toujours égale à 1.

4.3. Construction d'un classificateur via un algorithme d'exploration de données

À partir des métriques calculées pour chaque table, nous avons produit un jeu de données utilisable par un algorithme de datamining. L'objectif est de déduire, à partir des métriques, un modèle permettant la détection des tables de nomenclatures. L'outil Weka (Witten *et al.*, 2016) a été utilisé pour créer un arbre de décision implémentant ce modèle. Nous avons arrêté notre choix sur l'algorithme C4.5 (Quinlan, 2014)

construisant un arbre de décision qui fournit, dans notre cas, le meilleur compromis précision/rappel. Il gère les valeurs inconnues utilisées pour les métriques d'évolution des tables non modifiées durant la période d'observation. L'algorithme retourne une précision de 88,6% et un rappel de 88,7%. Durant l'expérimentation, il apparaît que C4.5 se base uniquement sur 2 des 5 propriétés: le nombre de clés étrangères et les contraintes d'intégrité référentielle (figure 4). Il ne prend pas en compte les métriques concernant l'historique des données. Notre hypothèse est que le nombre d'extractions de la base de données mis à notre disposition est insuffisant.

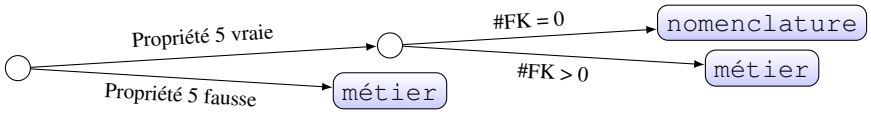


Figure 4. Arbre de décision généré par C4.5.

5. Travaux connexes

Il n'y a pas, à notre connaissance, de travaux s'étant intéressé au concept de table de nomenclatures dans la littérature scientifique. De même, nous n'avons pas trouvé de références concernant l'analyse de l'évolution des données contenues au sein d'une base de données relationelle en fonction du temps.

Dans le domaine de l'application de méthodes d'ingénierie logicielle aux bases de données relationnelles, des recherches ont déjà été effectuées sur l'évolution du schéma SQL. Hick et Hainaut (Hick, Hainaut, 2006) proposent DB-MAIN, une approche pour modéliser l'évolution d'une base de données à l'aide de transformations du schéma. Curino et al. (C. A. Curino *et al.*, 2008 ; C. Curino *et al.*, 2013) ont développé PRISM, un projet pour développer des méthodes d'automatisation de l'évolution d'un schéma. Celui-ci a été testé sur l'historique de MediaWiki (base de données de Wikipédia. Ce domaine de recherches est différent du sujet traité par ce papier. Nos analyses se fondent sur la structure des bases de données (e.g. nombre de tables, structures, types de relations entre les tables, types de contraintes appliquées sur les colonnes, etc...) et les informations concernant l'évolution des données.

La détection de problèmes de qualité dans une base de données a été abordée par Delplanque et al. (Delplanque *et al.*, 2017) avec DBCritics, un outil construisant un modèle de la base de données à partir du schéma SQL et vérifiant que celui-ci respecte un ensemble de règles prédéfinies. Sharma et al. (Sharma *et al.*, 2017) proposent DbDeo qui implémente la détection de 13 problèmes de qualité dans une base de données. La notion de table de nomenclatures pourrait être utile pour la détection de certains problèmes de qualité tel que l'usage abusif de littéraux dans du code SQL.

6. Conclusion

Dans cet article, nous avons déterminé quatre propriétés calculées à partir de la structure d'une base et une cinquième à partir de son historique de données. Chaque

propriété a été étudiée sur un cas d'étude afin de justifier sa pertinence. Pris séparément, ils s'avèrent tous insuffisants pour distinguer les tables de nomenclatures. Une utilisation conjointe de ces critères semble donc nécessaire. Leur complémentarité a été observée à l'aide d'un algorithme d'exploration de données (C4.5) mais celui-ci n'utilise que 2 des 5 critères pour classer les tables. Cela pourrait être dû aux données extraites de notre cas d'étude qui sont trop spécifiques ou parce que certains critères sont superflus. Néanmoins, le classificateur fournit de bons résultats et nous permet d'envisager une extension à cette étude sur un plus grand nombre de base de données.

Cet article ouvre plusieurs possibilités de travaux futurs. Tout d'abord, étant donné la difficulté à trouver une base de données dont le schéma SQL est consultable et accompagné d'un historique des informations sur plusieurs mois, il n'a pas été possible de reproduire l'expérience sur d'autres bases. Afin de confirmer la validité de la définition proposée, notre objectif est de diffuser un outil logiciel permettant aux architectes d'appliquer nos critères de classification des nomenclatures sur leurs bases de données (propriétés 1, 2, 3 et 5) et de nous transmettre le résultat validé table par table par leur soin. L'outil apportera une plus-value aux utilisateurs en produisant un rapport d'analyse (tables sans clé primaire, absence d'indexation, etc.) justifiant son usage. Par la suite, une analyse des données historiques de la base de données sur une plus grande période (propriété 4) serait souhaitable afin d'étudier un plus grand nombre de tables au sein de la base de donnnées. Le temps requis pour observer au moins une évolution sur chaque table de la base de données dépend de l'utilisation qui en est faite. Il sera également intéressant d'observer la déviance du contenu d'une même nomenclature entre différentes instances d'une même base de données.

Bibliographie

Curino C., Moon H. J., Deutsch A., Zaniolo C. (2013, février). Automating the database schema evolution process. *The VLDB Journal*, vol. 22, n° 1, p. 73–98.

Curino C. A., Moon H. J., Zaniolo C. (2008). Graceful database schema evolution: the prism workbench. *Proceedings of the VLDB Endowment*, vol. 1, n° 1, p. 761–772.

Delplanque J., Etien A., Auverlot O., Mens T., Anquetil N., Ducasse S. (2017). CodeCritics applied to database schema: Challenges and first results. In *Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference on*, p. 432–436. IEEE.

Hick J.-M., Hainaut J.-L. (2006). Database application evolution: a transformational approach. *Data & Knowledge Engineering*, vol. 59, n° 3, p. 534–558.

Le nouveau petit robert de la langue française 2009. (2008). Le Robert.

Quinlan J. R. (2014). *C4. 5: programs for machine learning*. Elsevier.

Sharma T., Fragkoulis M., Rizou S., Bruntink M., Spinellis D. (2017). Smelly Relations: Measuring and Understanding Database Schema Quality.

Witten I. H., Frank E., Hall M. A., Pal C. J. (2016). *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann.